**Basic SAS Skills**

**Handout #1 – In depth with PC SAS / “Remote Submit,” Macros, and Data Sources**

**Summer 2018**

**by John L. Campbell | Updated by S. J. Lusch & P. C. Machado & N. S. Seijas**

**TOPIC 1 – PC SAS CONNECT/ Remote Submit**

1. **PC SAS CONNECT**
   1. Relies on your PC getting an internet connection to the WRDS server.
   2. Advantages:
      1. Download data to your desktop and work with it there;
      2. Easier to edit your code because you can submit one statement at a time, log is color coded for your mistakes;
      3. View each dataset / output as you go along;
      4. Submit work to the server and still use your PC for other tasks; and
      5. If on a laptop, can work with SAS from anywhere (don’t have to be on campus).
   3. Disadvantages (there are ways around each of these):
      1. Space limitations of really large datasets on your PC;
      2. If you get disconnected before downloading data, must restart; and
      3. Some operations that require massive amounts of data to downloaded may “stall out.”
2. **Useful Instructions to start each program**
   1. WRDS Login:   
      Submitting this code will prompt a window asking for your WRDS ID and password:

%let wrds = wrds-cloud.wharton.upenn.edu 4016;  
options comamid = TCP remote=WRDS;  
signon user=\_prompt\_;

Submitting this code will keep your WRDS ID and password within your code and sign on for you:

%let wrds = wrds-cloud.wharton.upenn.edu 4016;  
options comamid = TCP remote=WRDS;  
signon user= 'XXXXXXXXXXXX' password= 'XXXXXXXXXXXX';

* 1. Library references, if any (designate your desired folder location):

libname home 'C:\SAS Library\SASCamp';

* 1. Macro references, if any (reference wherever you have saved your macro files):

\* Reference macros for use later;  
%include "C:\SAS Library\macros.sas";

* 1. Global SAS Commands – These are options that control the way SAS performs operations

options errors=**3** ls=**78** msglevel=i nocenter nodate noovp ps=max source;  
options mprint symbolgen;

|  |  |
| --- | --- |
| **SAS System Options** | |
| errors=3 | Set the number of observations that SAS reports as errors to 3 |
| ls=78 | Specifies the line size (printer line width) in characters for the SAS log and the SAS output that are used by the DATA step and procedures |
| msglevel=i | Print additional notes in the SAS Log |
| nocenter | Left aligns SAS procedure output |
| nodate | Prevents the date and time from printing on the top of each page of the SAS log |
| noovp | Prevents SAS from emphasizing error messages |
| ps=max | Sets the number of lines that compose a page to the maximum setting, 32,767 |
| source | Specifies to write SAS source statements to the SAS log |
| **SAS Macro Options** | |
| mprint | Displays the SAS statements that are generated by macro execution |
| symbolgen | Displays the results of resolving macro variable references |

1. **Once logged into WRDS, use the RSUBMIT command.**
   1. RSUBMIT (or “remote submit”) is the command that lets the program know that you are now working on the WRDS server rather than your PC. An error message will occur if you haven’t successfully logged on to WRDS. If this occurs, simply rerun the login code.

rsubmit;

* 1. At the end of every rsubmit block remember to use the run instruction to tell SAS to execute the remote section.

run;

* 1. ENDRSUBMIT lets the program know that you done working on WRDS, and are now working on your PC.

endrsubmit;

* 1. This is a simple example of the usage of SAS CONNECT (from the WRDS articles). It reads the annual fundamentals dataset of Compustat, selects the 2010 fiscal year, and prints the first 30 lines of the output.

%let wrds=wrds-cloud.wharton.upenn.edu 4016;

options comamid=TCP remote=WRDS;

signon username=\_prompt\_;

rsubmit;

**data** mydata;

set comp.funda;

where fyear=2010;

**proc** **print** data=mydata (obs=30);

**run**;

endrsubmit;

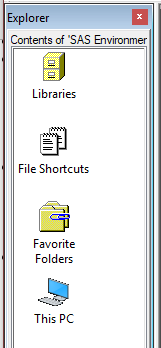
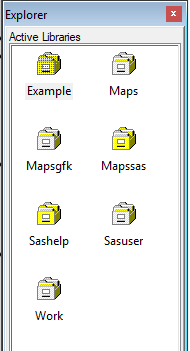
1. **WARNING: Always know where your data is!**
   1. The biggest difference between PC SAS/Remote submit vs. UNIX is keeping track of where your data is when writing commands.
   2. If you “ENDRSUBMIT;” without downloading the data to your desktop, it’s left on the WRDS server. You will almost always want to use PROC DOWNLOAD before the ENDRSUBMIT command:

**proc** **download** data=notredame#1; **run**;

* 1. Suppose you downloaded your dataset and manipulated the data to show how awesome Notre Dame football recruiting was in 2012. The new dataset is named “notredamepreseason#1.”  
     Now you want CRSP data on how much market capitalization USC and Michigan lost during the announcement periods when Notre Dame selected their awesome recruits. Unless you upload the new dataset to the WRDS server, using RSUBMIT to merge the CRSP data with the new dataset won’t work. Use the PROC UPLOAD command:

**proc** **upload** data=notredamepreseason#1; **run**;

1. **Monitor your data.**
   1. A major benefit of a dataset downloaded to your desktop is being able to open it and observe data output to ensure it’s doing what you want.
   2. To open a dataset that is downloaded onto your PC:
      1. Highlight the “Explorer” tab;
      2. Double click on “Libraries;”
      3. Double click on the library that contains your dataset (if you did not specify a library then the data will be in “Work”); and
      4. Find the name of the dataset you want, and double click on it.



**CONGRATULATIONS! Now you are ready to use PC SAS/Remote Submit! GOOD LUCK…**

**Digression on Libraries – Another Approach**

To set up libraries without using the libname function inside of your code, you can set up “permanent libraries.” First create the data folder in windows, and then right-click in the “Active Libraries” window, select “New”, check the box “Enable at startup,” name the library, and then click on the windows folder as your library location. You can then save datasets to the library with the following command:

**data** notredame.underachievers;

set underachievers; **run**;

And now dataset “underachievers” has been saved to the notredame library.

**TOPIC 2 - Macros**

Macros are designed to automate repetitive tasks. You can write a macro within the source code, or you can reference macros written in another SAS file. For the purposes of this course, we will not confuse you with writing your own macros. Rather, we just want you to be aware of the macros that do exist and how to use them if you want.

**WARNING: You should only use macros if you are completely comfortable with the underlying code. As a result, when first starting out it may be wiser to do your own coding for most of these tasks. If you do use one of these macros, ALWAYS open the dataset and ensure that it did what it was supposed to do!**

The macros currently available are as follows:

1. LAGS – lag a variable by month, quarter or year
2. LAGSYA – lag a variable using “yeara” variable in compustat
3. WT – winsorize or truncate specified variables in your dataset
4. WTSTD – winsorize some and truncate other variables in the same dataset
5. FMB – Fama-Macbeth (1973) regression estimates
6. CORRPS – outputs a correlation matrix with pearson on top, spearman (rank) on bottom
7. FF17 – creates Fama-French 17 industry classifications within dataset
8. FF30 – creates Fama-French 30 industry classifications within dataset
9. FF48 – creates Fama-French 48 industry classifications within dataset
10. FF49 – creates Fama-French 49 industry classifications within dataset....same as FF48 except a new class is created for gaming/computer software
11. BBL\_IND – creates 15 industry classifications based on BBL(1998)
12. IND22 – creates 22 industry classifications based on Barth et al. paper
13. DUMMY - turn a single variable into many dummy variables
14. Demean\_x1 - demean data along one dimension (i.e. Industry)
15. Demean\_x2 - demean data along two dimensions (i.e. industry AND year)
16. Clus2d - 2 Way Clustering Macro (Gow, Ormazabal, Taylor 2010 TAR)
17. VUONG - computes Vuong (1989 Econometrica) test of two non-nested regressions
18. CRAMER1 – similar to Vuong Z-stat, compares R2 of two non-nested models
19. robust\_hb - calculates iteratively weighted least squares standard errors

To invoke a macro within your code, you need to use “%” and then the macro name. For instance, to create Fama-French 48 industry codes you only have to use one line of code:

%***FF48***(**data**=forecast2, newvarname=industry, sic=sic, out=forecast2);

One key whenever you use macros is to read about the necessary inputs to the macro code. This is done within the macro .sas file. As another example, see the heading information from the LAGS file:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* CREATED BY: Ryan Ball (UNC-Chapel Hill) \*/

/\* MODIFIED BY: Scott Dyreng (UNC-Chapel Hill) \*/

/\* DATE CREATED: November 29, 2005 \*/

/\* LAST MODIFIED: November 29, 2005 \*/

/\* MACRO NAME: Lags \*/

/\* ARGUMENTS: 1) DATA: input dataset containing variables that will be lagged \*/

/\* 2) FIRMID: firm-specific identification variable \*/

/\* 3) TIMEID: date variable (e.g fyenddt, fqenddt) \*/

/\* 4) VARS: variable(s) that will be lagged..macro will automatically add a \*/

/\* '\_lead#' or '\_lag#' suffix to all lead/lag variables \*/

/\* 5) LAG\_TYPE: = 1 if monthly data is used \*/

/\* = 3 if quarterly data is used \*/

/\* = 12 if yearly data is used (default) \*/

/\* 5) NUM\_LAGS: number of lags taken...positive values represent lags of each\*/

/\* variable while negative value represent lead values \*/

/\* 6) OUT: = output dataset with orinal plus lagged variables \*/

/\* DESCRIPTION: This macro takes the input variables and computes lagged variables based \*/

/\* on other specifications (e.g. FIRMID, TIMEID, LAG\_TYPE) \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

So when you invoke the LAGS macro, you will want to make sure you specify these six arguments in your code before it will work. An example code:

%***lags***(data=qtrearn, firmid=gvkey, timeid=date, vars=mve data25, lag\_type=**3**, num\_lags=**1**, out=qtrearn1);

**TOPIC 3 – Data Sources**

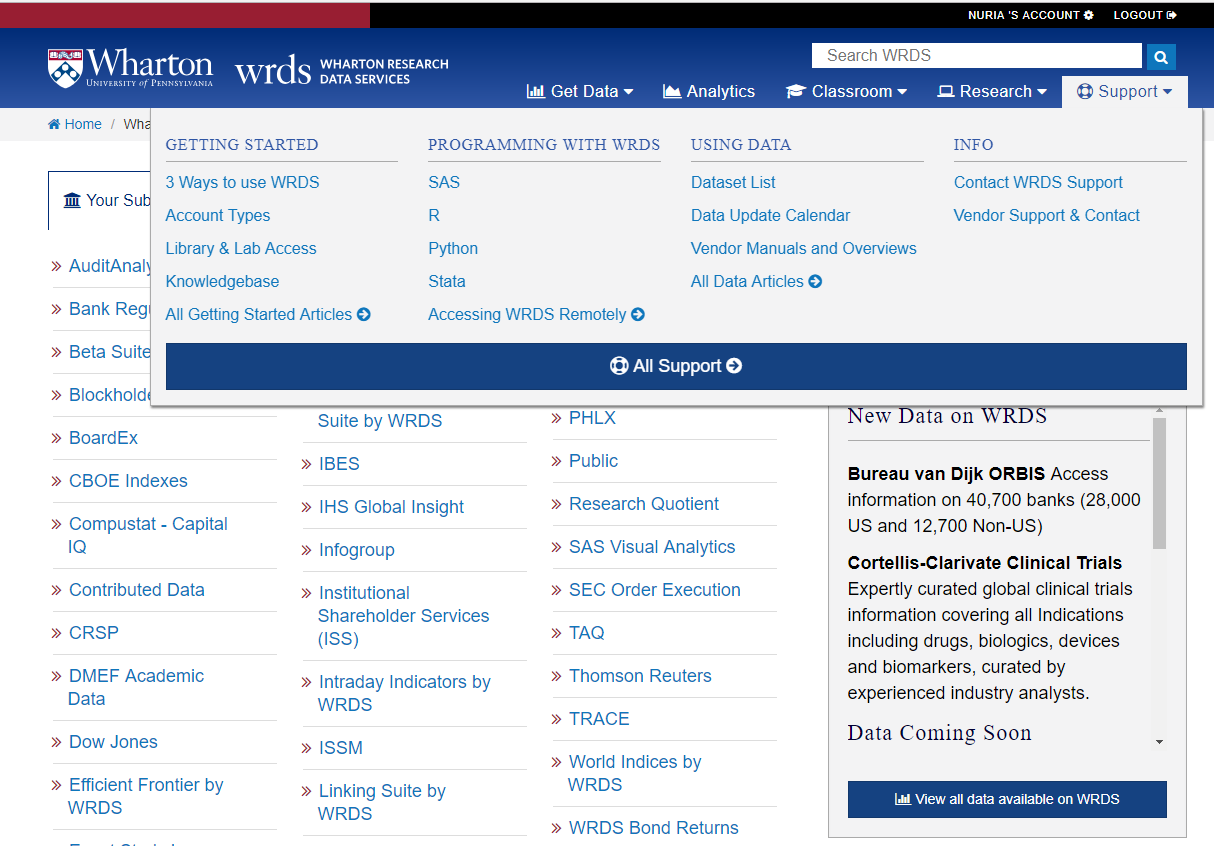
WRDS is SAS powered. That is, all WRDS databases are maintained in SAS and the data processes that underlie web access to WRDS libraries are programmed in SAS. A SAS data library is a directory where SAS data sets are saved. A libname, or libref (library reference), is a user-defined name that refers to a SAS data library. For example, the libref CRSP refers to all SAS data sets in the UNIX directory /wrds/crsp/sasdata/sm.

What is the easiest way to know which databases you want to pull data from?

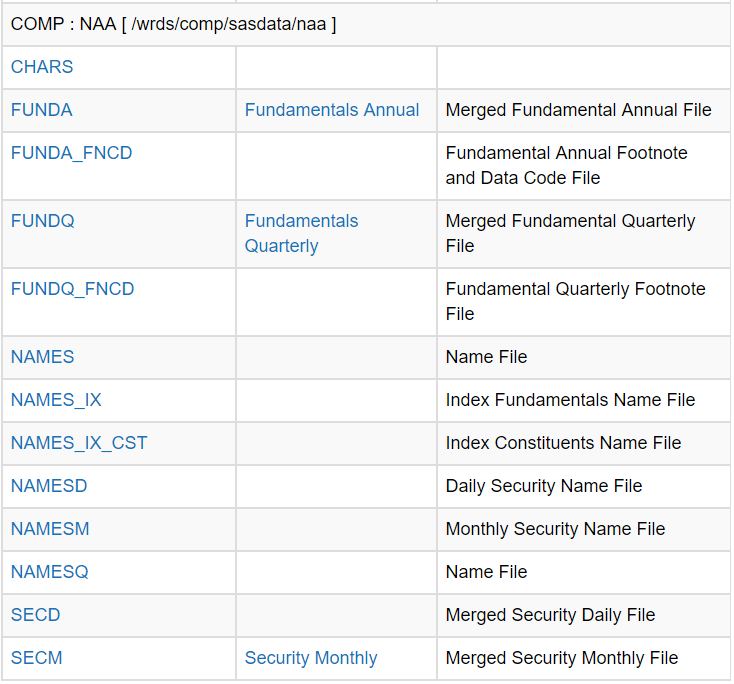
* Logon to the WRDS website: <https://wrds-web.wharton.upenn.edu/wrds/>.

The “HOME” page, useful for two primary tasks:

* WRDS alerts are given at the top of the front page. The WRDS alert system is used to make users aware of database issues, database updates, new datasets, training sessions on databases, etc...
* On the left hand side of the front page you can see the current databases subscriptions.



The “SUPPORT” tab:

* Click on the “Dataset List” link under “Using Data.” You will see a list of all the datasets on the WRDS server, including those that we do not have a subscription for. In order to see which databases are available in a dataset, click on an available dataset (e.g. Compustat). Here is an example of a portion of the databases you will find in COMPUSTAT.

Clicking on a database will list all of the variables within that dataset. This is the BEST way to see what information is in a dataset and how to get that information into a dataset for you to use in data analysis.

For example, click on FUNDA and you will see the following:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Position** | **Variable Name** | **Type** | **Length** | **Description** |
| 1 | gvkey | Char | 6 | Global Company Key |
| 2 | datadate | Num | 8 | Data Date |
| 3 | fyear | Num | 8 | Data Year - Fiscal |
| 4 | indfmt | Char | 12 | Industry Format |
| 5 | consol | Char | 2 | Level of Consolidation - Company Annual Descriptor |
| 6 | popsrc | Char | 1 | Population Source |
| 7 | datafmt | Char | 12 | Data Format |
| 8 | tic | Char | 8 | Ticker Symbol |
| 9 | cusip | Char | 9 | CUSIP |
| 10 | conm | Char | 57 | Company Name |

This is a list of the first 10 variables of the 1,000+ variables available in the FUNDA database. Note

* Variable Name – The name used by WRDS, and what you will reference to retrieve the data
* Type – Identifies whether the variable is stored in character or numeric format.
  + Usually, character variables will be made up of letters while numeric will use numbers, but numeric variables can sometimes be classified as characters (such as when you import external data into SAS). See The Little SAS Book, pages 292-293 for the commands to change a character variable to a numeric variable and vice versa.
* Length – The character length of the data. Mainly a concern when you are trying to merge datasets (e.g. Cusip is defined as either 6-digits or 8-digits depending on the dataset).
* Description – Brief description of the variable.

Notice that all databases follow a naming convention. In particular, each database name has a first and last name, such as comp.funda. The “first” name lets WRDS know which UNIX Directory to look in.

Common directory “first” names are as follows:

|  |  |
| --- | --- |
| **Data Source** | **“First” name / UNIX Directory** |
| Compustat | comp |
| CRSP | crsp |
| I/B/E/S | ibes |
| Audit Analytics | audit |
| Execucomp (is within Compustat) | comp |
| Thomson Reuters, for institutional ownership | tfn |

And common “second” names:

COMPUSTAT Merged Fundamental Annual File 🡪 .funda

COMPUSTAT Merged Fundamental Quarterly File 🡪 .fundq

EXECUCOMP Annual Compensation 🡪 .anncomp

CRSP daily stock returns file 🡪 .dsf

CRSP daily stock index file 🡪 .dsi

CRSP monthly stock returns file 🡪 .msf

CRSP monthly stock index file 🡪 .msi

Audit analytics audit fee data file 🡪 .auditfees

I/B/E/S detail history dataset 🡪 .det\_epsus

I/B/E/S summary history dataset 🡪 .statsum\_epsus

Thomson-Reuters Institutional Ownership 13F Holdings 🡪 .s34

Thomson-Reuters DealScan🡪 .dealscan

The two most common ways to get data from these sources is to use either a DATA step or a PROC SQL step.

1. DATA step:  
   For example, you want to pull from WRDS a list of all companies included in the database from years 2002 - 2009 with total assets greater than $100,000. The variables you wish to pull down are gvkey, company name, cusip, ticker, SIC, year, total assets, net income, shares outstanding, closing share price at the companies' fiscal year.

rsubmit;

**data** firms;

set comp.funda (keep= gvkey conm cusip tic sich fyear at ni csho prcc\_f indfmt datafmt popsrc consol);

where (fyear between **2002** and **2009**)and (at > **100000**);

if indfmt='INDL';

if datafmt='STD';

if popsrc='D';

if consol='C';

**run**;

**proc download** data = firms;

**run**;

endrsubmit;

1. PROC SQL step:  
   For example, assume you now want to take your dataset from step 1 and add to that audit fee data from audit analytics. (NOTE: requires that original dataset is on the WRDS server, i.e. PROC UPLOAD)

rsubmit;

**proc** **upload** data = firms;

**run**;

**proc** **sql**;

create table firmsaudit

as select a.\*, b.audit\_fees

from firms as a, audit.auditfees as b

where (a.tic = b.best\_edgar\_ticker) and (a.fyear = b.fiscal\_year);

**quit**;

**proc** **download** DATA = firmsaudit;

**run**;

endrsubmit;